WEST

End of Result Set

Generate Collection | Print

L8: Entry 1 of 1 File: USPT Jun 9, 1998

DOCUMENT-IDENTIFIER: US 5765168 A
TITLE: Method for maintaining an index

Brief Summary Text (2):
This invention relates generally to maintaining an index of a database, and more particularly to deleting entries from the index as records of the database are deleted.

Drawing Description Text (12):
FIG. 11 is a block diagram of a remapping table used while deleting entries;

Drawing Description Text (26):
FIG. 25 is a flow diagram of a process for deleting pages; and

Detailed Description Text (120):
Deleting a single page may require the reordering of millions of location values of the data structures of the index 70 of FIG. 9 because of "holes" left by deleted words and location entries. For any page which is deleted, all of the locations of the following pages need to be adjusted, byte by byte. For example, if a deleted page includes 888 words, the locations of the following pages need to be reduced by 888.

Detailed Description Text (125):
By partitioning the index 70 over the multiple data structures 1001, the updating problems stated above are minimized since the size of the files concurrently being modified is greatly reduced. Multiple files allow small changes to be made to the index 70 without incurring too much additional maintenance overhead.

Detailed Description Text (136):
Deleting Entries

Detailed Description Text (137):
During merge/sort, deleted entries of the index are expunged. The deleting of entries proceeds as follows. Remember, all words and metawords and their locations are sequentially indexed. Therefore, deleting a page can affect a large portion of the index 70.

Detailed Description Text (363):
Deleting Duplicate Pages

Detailed Description Text (364):
FIG. 25 shows a process 2500 for deleting pages. In step 2510, determine if the page to be deleted is a master page. If true, then generate a reissue request 2521 for the automated browser 20 of FIG. 2 using the address (URL) of the next recorded duplicate page. Then, in step 2530, promote the next duplicate page to be a master page. In step 2540, generate a "deleted" metaword pair [location, .quadrature.DELETED.quadrature.], where location is the location of the last word of the page to be deleted. Otherwise, if false, e.i, this is not a master page, then in step 2550, determine if the page to be deleted is the next recorded duplicate of the page to be deleted and there is no master page, proceed with step 2520 if true. Otherwise, if false, proceed with step 2540.

Current US Original Classification (1):
707/200

CLAIMS:

2. The method of claim 1 further comprising:

writing a deleted location entry in an index entry having a word entry indicating a deleted status of a deleted record identified by the deleted location entry;

deleting the location entries of the deleted record substantially simultaneous with merging the tiers of files.

L7: Entry 5 of 6 File: USPT Jan 31, 1995

DOCUMENT-IDENTIFIER: US 5386553 A
TITLE: Disk file updating control device and method using updating data stored in a first-in-first-out queue

Abstract Text (1):
In a disk file control device in which updating data output by a data processing section is temporarily stored in a FIFO queue to update a file, the updating data has a data structure which includes an operation code representing a type of updating operation such as a data inserting, removing or updating operation, and amount-of-data information indicating an amount of data to be subjected to the type of updating operation. The type of operation is determined from a given operation code and is used to change the size of file to be updated by as much as an amount of data obtained from the amount-of-data information, and thereafter the data stored in the FIFO queue is transferred into the file thus changed. Alternatively, the type of operation is used to change the size of the original file to be updated to provide a new file, and the data of the original file and the FIFO queue are transferred into the new file.

Brief Summary Text (16):
As a result, the above-described conventional technique can only perform an updating operation with the file remaining fixed in size. In other words, the conventional technique is applicable only to an updating operation which is not accompanied by data insertion or removal.

Brief Summary Text (22):
To achieve the objects and in accordance with the purposes of the invention, as embodied and broadly described herein, there is provided a disk file control device in which updating data output by a data processing section is stored in an FIFO queue including a nonvolatile memory, and is used to update a file. The updating data has a data structure including an operation code representing a type of updating operation such as a data inserting, removing or updating operation, amount-of-data information indicating an amount of data to be subjected to the type of updating operation, and a file page containing data for writing into the disk file for updating. The disk file control device includes means, operatively coupled to the FIFO queue, for determining a type of updating operation according to the operation code in the data structure; means, operatively coupled to the type determining means and the disk file, for changing the size of the disk file according to the determined type of updating operation, and by the amount obtained from the amount-of-data information; and means, operatively coupled to an output of the changing means and to the FIFO queue, for reading out the data stored in the file page of the FIFO queue into the disk file after the size of the file has been changed.

Brief Summary Text (23):
There is also provided in accordance with the invention a disk file control device in which updating data output by a data processing section is stored in an FIFO queue including a nonvolatile memory, and is used to update a disk file. The updating data has a data structure including an operation code representing a type of updating operation such as a data inserting, removing or updating operation, amount-of-data information indicating an amount of data to be subjected to the type of updating operation, and a file page containing new data for writing into the disk file for updating. The disk file control device includes means, operatively coupled to the FIFO queue, for determining the type of updating operation according to the operation code in the data structure; means, operatively coupled to the type determining means and the disk file, for defining a new disk file, said new disk file having a size

substantially equal to that of an original disk file changed according to the determined type of updating operation and by an amount determined from the amount-of-data information; and means for transferring the data of the original file and the file page of the FIFO queue into the new file.

Brief Summary Text (24):
Further in accordance with the invention there is provided a method for performing an updating operation of a disk file in accordance with updating data output by a data processing section, the updating data being a data inserting, removing or updating operation. The data processing section stores updating data in a FIFO queue, the updating data having a data structure including an operation code, amount-of-data information and a file page. The operation code represents the type of updating operation to be performed, the amount-of-data information represents an amount-of-data to be subjected to the updating operation, and the file page contains data for writing into the disk file for updating. The method includes the steps of determining in accordance with the operation code the type of data updating operation to be performed; changing a size of the disk file according to the type of updating operation to be performed and the amount of data represented by the amount-ofdata information; and reading out the data stored in the file pages of the FIFO queue for writing into the disk file.

Detailed Description Text (26):
In the above-described first embodiment, after the size of the file is changed, the operation code is set to "updating" because of the following reason. If the operation code remains as "inserting" or "removing", then when a system interruption occurs after the file size is changed, the file size would, incorrectly, be changed again according to the operation code after the system is reactivated. This difficulty is overcome by changing the operation code to the "updating" operation code after the change of the file size.

Current US Original Classification (1):
707/205

WEST

Generate Collection | Print

L7: Entry 4 of 6 File: USPT May 7, 1996

DOCUMENT-IDENTIFIER: US 5515532 A
TITLE: File management system for memory card

Detailed Description Text (67):
When the writing of definition information is successfully completed (YES in step ST44), the used sizes of all data files from a file looked up in the current file size checking operation to the current file are updated (step ST48). Upon updating, of the above-mentioned data file management information definition word (FIG. 4B), a US part is updated, and BCC2 is calculated from values DFST and US, thus simultaneously updating BCC2. At this time, if updating of US and BCC2 is not successfully completed (NO in step ST50), response data indicating a writing error is output (step ST46), and the control returns to a command wait state (START in FIG. 7A).

Current US Original Classification (1):
707/200

WEST

Generate Collection | Print

L9: Entry 1 of 3 File: USPT Aug 27, 1996

DOCUMENT-IDENTIFIER: US 5551027 A
TITLE: Multi-tiered indexing method for partitioned data

Detailed Description Text (25):
With respect to record selectivity, it should be noted that if the average number of index entries for a Key Value in a Local Index is n, the probability that a Local Index update operation is either inserting a new Key Value or deleting the last instance of a Key Value, thereby causing a Global Index insertion or deletion, is usually much smaller than 1/n. For example, consider an index on the DEPARTMENT field of an EMPLOYEE table. The index is updated whenever a person joins the company, changes his/her department, or leaves the company. However, except for the first person joining a new department and for the last person leaving a department (e.g., when a department is dissolved), all other personnel changes at a worksite do not cause a Global Index update regardless of the average size of a department.

L10: Entry 3 of 24 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5873097 A
**** See image for Certificate of Correction ****
TITLE: Update mechanism for computer storage container manager

Brief Summary Text (25):
Another conventional technique for handling atomic updates is sometimes referred to as the "shadow page technique". In this technique, the base file is divided into pages, and an index to the current version of the pages is maintained. Updates are managed atomically at the page level rather than the file level, and are accomplished by writing the new version of a page, then updating the index to point to the new version rather than the old version of that page, and then deleting the old version of the page. In some implementations, the index itself may also be shadowed. The set of old pages, identified by the old index, and the set of new pages identified by the new index, completely describe the base and updated states of the information, respectively.

Brief Summary Text (35):
The changes which are identified in an update container, if they represent modifications to an object in an underlying container, refer to that object logically rather than physically. For example, they include a reference to a persistent object identifier which is unique within the underlying contianer and its own base containers. Thus certain changes are allowed on existing closed containers, such as rearranging information or deleting superfluous information, as long as the logical view of the closed container remains the same.

Detailed Description Text (709):
3. Data editing (comprising inserting, deleting, and overwriting of data)

Detailed Description Text (711):
5. Deleting a value

Detailed Description Text (712):
6. Deleting a property

Detailed Description Text (713):
7. Deleting an object

Detailed Description Text (757):
Deleting data

Detailed Description Text (758):
There are four cases that can occur when deleting data. Unlike inserts, multiple segments can be involved. When a CMDeleteValueData() is done, a starting offset and size are passed. The size may span from some portion within a segment, through any number of whole segments, and end somewhere in a final segment. Alternatively, the size may specify only a portion of a single segment. However, each of these processes can be viewed separately in terms of a single segment. Thus, for any one segment, the following four cases are possible:

Detailed Description Text (759):
Case (1): Deleting an entire segment.

Detailed Description Text (760):

Case (2): Deleting the left portion of a segment.

Detailed Description Text (761):
Case (3): Deleting the right portion of a segment.

Detailed Description Text (762):
Case (4): Deleting an interior portion of a segment.

Detailed Description Text (782):
Deleting a value

Detailed Description Text (783):
Deleting a value involves freeing all the value segment memory. The value header itself is moved to a "deleted values" list and never freed. This is done to make sure the user does not try to reuse a value refNum for a deleted value. Value headers on the "deleted values" list are all flagged as "deleted" so that this check can be done.

Detailed Description Text (788):
Deleting a property

Detailed Description Text (789):
Properties can be deleted in two possible ways; (1) implicitly by deleting all the values for a property, and (2) explicitly doing a CMDeleteObjectProperty(). For updating purposes, only the explicit case need be considered. The implicit deletion will happen as a byproduct of value deletions.

Detailed Description Text (794):
Implicit value deletion is essentially a subset of the explicit deletion case. The difference between deleting an individual value, and deleting a value when its property is going to be deleted, is that a "deleted value" entry is created in the individual delete case, while the touched list entry is always freed in the property value delete case. What both have in common, however, is that "removed" entries in other objects for values that were moved into the property being deleted are set to "deleted value".

Detailed Description Text (795):
Thus all the touched list entries for the values belonging to the property are deleted from the touch list. Any values moved into this object from other objects also have a "removed" touched list entry in their original object. Deleting such values for the property has exactly the same effect as explicit value deletion, i.e., the "removed" entry is changed to a "deleted value" entry in the original object. The "inserted" entry is, of course, removed.

Detailed Description Text (797):
Deleting an object

Detailed Description Text (798):
Deleting an object means all its values and properties are deleted. Objects can only be explicitly deleted using CMDeleteObject(). Deleted objects are placed on the "deleted objects" list and marked deleted to prevent the user from further use of them.

Detailed Description Text (848):
Close time processing generates two separate groups of updating instructions: those updating all the values for a single object and those for deleting objects and properties.

Detailed Description Text (874):
Repeated calls to these handlers are done to manipulate the updating instructions. It's not the most efficient mechanism, but as previously stated, it's a compromise. To aid in the efficiency, buffered I/O is done. Note that updating instructions are all value data for a single value: a value for the special updating property of an object to be updated, or a value for the special TOC #1 property. The data is actually read or written using the standard API calls to CMReadValueData() and CMWriteValueData().

Thus the updating code controls the buffer, its size, and the starting offset; the parameters to the CMReadValueData() and CMWriteValueData().

WEST

Generate Collection | Print

L13: Entry 5 of 14 File: USPT Feb 1, 2000

DOCUMENT-IDENTIFIER: US 6021415 A
**** See image for Certificate of Correction ****
TITLE: Storage management system with file aggregation and space reclamation within aggregated files

Detailed Description Text (68):
When the predetermined criteria are satisfied, step 608 creates a managed file from the user files meeting the criteria. This step is performed by updating the database 113 to recognize the constituent user files (meeting the criteria) as a single managed file. In particular, the subsystem 102 in step 608 enters a representation of the newly created managed file and its constituent user files in the database 113. This involves entering: (1) the user files in the inventory table (e.g., Table 1,shown above), (2) the managed file in the storage table (e.g., Table 2, shown above), (3) the managed file and its constituent user files in the mapping table (e.g., Table 3, shown above), and (4) the managed file in the managed file attributes table (e.g., Table 4, shown above).

Detailed Description Text (111):
This operation involves moving an entire managed file from one location to another in the storage hierarchy 114, and updating the database 113 accordingly. This operation is a necessary part of other operations, such as migration, reclamation, etc. Advantageously, managed file move involves significantly reduced the management file overhead costs, due to the treatment of all constituent user files as a single aggregate file.

Detailed Description Text (160):
After step 906, step 908 determines whether the data storage unit's computed efficiency rating passes a predetermined threshold. This threshold is set according to the user's specifications, e.g. 40% or 60%. If the efficiency rating exceeds the threshold, the data storage unit would not benefit sufficiently from reclamation. In this case, step 910 waits for one of the conditions discussed above (step 902) before restarting step 906. Whenever the efficiency rating fails the threshold, however, reclamation operations commence in step 912.

Current US Original Classification (1):
707/206

Current US Cross Reference Classification (1):
707/205

CLAIMS:

16. A method of reclaiming a source storage area to a target storage area, said source storage area including one or more managed files, each managed file including one or more user files, said method comprising:

evaluating the source storage area by determining whether data storage efficiency therein exceeds predetermined criteria;

if the storage area fails the predetermined criteria, performing a reclamation process, said reclamation process comprising, for each managed file in the source storage area:

determining whether the managed file contains any deleted-file storage space;

in response to finding deleted-file storage space, performing a process of reconstructing the managed file comprising:

identifying contiguous regions of non-deleted user files within the managed file; and

copying each identified contiguous region to adjacent locations in a target storage area to form a reconstructed managed file.

48. A signal-bearing medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method for consolidating a unit of stored data from a source location to a target location, said unit including multiple managed files, each managed file including one or more user files, said method comprising:

evaluating the source storage area by determining whether data storage efficiency therein exceeds predetermined criteria;

if the storage area fails the predetermined criteria, performing a reclamation process, said reclamation process comprising, for each managed file in the source storage area:

determining whether the managed file contains any deleted-file storage space;

in response to finding deleted-file storage space, performing a process of reconstructing the managed file comprising:

identifying contiguous regions of non-deleted user files within the managed file; and

copying each identified contiguous region to adjacent locations in a target storage area to form a reconstructed managed file.

80. A data storage subsystem, comprising:

a storage hierarchy including a source location and a target location; and

a digital data processing apparatus coupled to the storage hierarchy;

wherein the digital data processing apparatus is programmed to perform a method for consolidating a unit of stored data from the source location to the target location, said unit including multiple managed files, each managed file including one or more user files, said method comprising:

evaluating the source storage area by determining whether data storage efficiency therein exceeds predetermined criteria;

if the storage area fails the predetermined criteria, performing a reclamation process, said reclamation process comprising, for each managed file in the source storage area:

determining whether the managed file contains any deleted-file storage space;

in response to finding deleted-file storage space, performing a process of reconstructing the managed file comprising:

identifying contiguous regions of non-deleted user files within the managed file; and

copying each identified contiguous region to adjacent locations in a target storage area to form a reconstructed managed file.

L13: Entry 10 of 14 File: USPT Jul 7, 1998

DOCUMENT-IDENTIFIER: US 5778393 A
**** See image for Certificate of Correction ****
TITLE: Adaptive multitasking for dataset storage

Detailed Description Text (15):
Next, task 308 asks whether the size of the selected dataset exceeds a predetermined threshold. If the dataset is too large, it may be more efficient to break the dataset into parts and store the parts concurrently on multiple devices 106a-106e. If the dataset's size exceeds the threshold, task 309 invokes an "oversize routine", which begins in task 402 of FIG. 4 (described below). Otherwise, query 308 directs control to task 310, which selects an available one of the devices 106a-106e to store the selected dataset. A device is "available" if it is not already busy storing another dataset. Available data storage devices therefore may also be referred to as "non-busy", with unavailable data storage devices being "busy". As discussed in greater detail below, when a device begins storing a dataset, the device may be marked, flagged, logged, or otherwise designated as unavailable.

Detailed Description Text (29):
After task 314, query 316 asks whether the size of the recently selected dataset exceeds the predetermined threshold. If the dataset's size exceeds the threshold, task 317 invokes the oversize routine at task 402 of FIG. 4 (described below). Otherwise, query 316 directs control to task 318, which selects the next available device, in the same manner as task 310. After task 318, control proceeds to tasks 320-324, for storage of the current dataset in the selected device as discussed above. Also after task 318, query 312 is repeated to determine whether the buffer contains any more datasets to process. If so, processing of the new dataset(s) is initiated; this may occur in parallel with the storage of one or more previous datasets, as discussed above.

Detailed Description Text (32):
As explained above in conjunction with FIG. 3 (tasks 309, 317), the oversize routine is performed if the size of an incoming dataset exceeds a predetermined threshold. FIG. 4 describes an exemplary sequence of tasks 400 illustrating one embodiment of oversize routine.

Current US Original Classification (1):
707/205

Current US Cross Reference Classification (1):
707/200

CLAIMS:

4. The method of claim 1, step (c) further comprising the steps of:

determining whether the selected dataset exceeds a predetermined size threshold, and

if so, dividing the selected dataset into a number of sub-components and

separately processing each sub-component as one of the plurality of datasets.

8. The method of claim 1, further comprising steps of:

creating a status record indicating whether each of the data storage devices is available or not available to satisfy data storage requests;

each time storage of a dataset begins in a data storage device, updating the status record to indicate that said data storage device is presently not available to satisfy data storage requests;

said step of employing a predetermined criteria to select an available data storage device comprising the steps of referencing the status record to identify a data storage device that is available to satisfy data storage requests.

18. The article of manufacture of claim 15, step (c) further comprising the steps of:

determining whether the selected dataset exceeds a predetermined size threshold, and if so, dividing the selected dataset into a number of sub-components and separately processing each sub-component as one of the plurality of datasets.

22. The article of manufacture of claim 15, further comprising steps of:

creating a status record indicating whether each of the data storage devices is

available or not available to satisfy data storage requests;

each time storage of a dataset begins in a data storage device, updating the status record to indicate that said data storage device is presently not available to satisfy data storage requests;

said step of employing a predetermined criteria to select an available data storage device comprising the steps of referencing the status record to identify a data storage device that is available to satisfy data storage requests.

32. The apparatus of claim 29, step (c) further comprising the steps of:

determining whether the selected dataset exceeds a predetermined size threshold, and if so, dividing the selected dataset into a number of sub-components and separately processing each sub-component as one of the plurality of datasets.

36. The apparatus of claim 29, further comprising steps of:

creating a status record indicating whether each of the data storage devices is available or not available to satisfy data storage requests;

each time storage of a dataset begins in a data storage device, updating the status record to indicate that said data storage device is presently not available to satisfy data storage requests;

said step of employing a predetermined criteria to select an available data storage device comprising the steps of referencing the status record to identify a data storage device that is available to satisfy data storage requests.